**A05:2021 – Security Misconfiguration**

**Summary:**

A security misconfiguration occurs when services are delivered with insecure default settings or when security choices are not defined in a way that maximizes security. This is possible across all computing platforms, software programs, cloud services, and network infrastructure. Few common mistakes are:

1. Unnecessary features are enabled or installed like ports, services and account.

2. Missing any component of the application stack without the necessary security protection.

3. Error handling reveals stack traces or other overly informative error messages.

4. The default accounts are still active, and their passwords are unchanged.

5. The software or applications are out of date or vulnerable.

6. Missing important upgraded systems or latest security features are disabled or not configured.

**Prevention**

OWASP recommendations to avoid such security misconfiguration issues are implementing repeatable, automated security hardening processes, segmenting the application architecture, and keeping the platform minimal or limited. Also sending the security directives to clients, e.g. security headers.

**A06:2021 – Vulnerable and Outdated Components**

**Summary:**

The vulnerable and outdated components refer to third-party libraries or frameworks used in web applications that have known vulnerabilities or are no longer supported by their developers. These components can be exploited by attackers to gain unauthorized access to sensitive data or take control of the system. Few common examples are:

1. Application or software is vulnerable, unsupported, or out of date.

2. Did not know the versions of all components you use in application and software.

3. Did not scan for vulnerabilities regularly and subscribe to security bulletins related to the components.

4. Not fix or upgrade the underlying platform, frameworks, and dependencies of the system.

5. Software developers do not test the compatibility of updated, upgraded, or patched libraries.

**Prevention**

To prevent the risks of using vulnerable and outdated components, it is very important to keep them up-to-date. This involves regularly checking for security updates or patches from the component's developers and applying them promptly. Underlying system and platform must be upgraded and keep them up to mark. Also monitor, detect or protect against the discovered issues.

**A03:2021 – Injection**

**Summary:**

Injection or SQL injection are the common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. Hacker use malicious SQL statements in the input box, and in response, the database presents sensitive information. Few examples are:

1. Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.

2. Data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.

3. User data is not validated, filtered, or scanned by the application.

4. Data is directly used or concatenated.

**Prevention**

To prevent from these injections developer must utilizing parameterized database queries with bound, typed parameters and careful use of parameterized stored procedures in the database. Some basic prevention is:

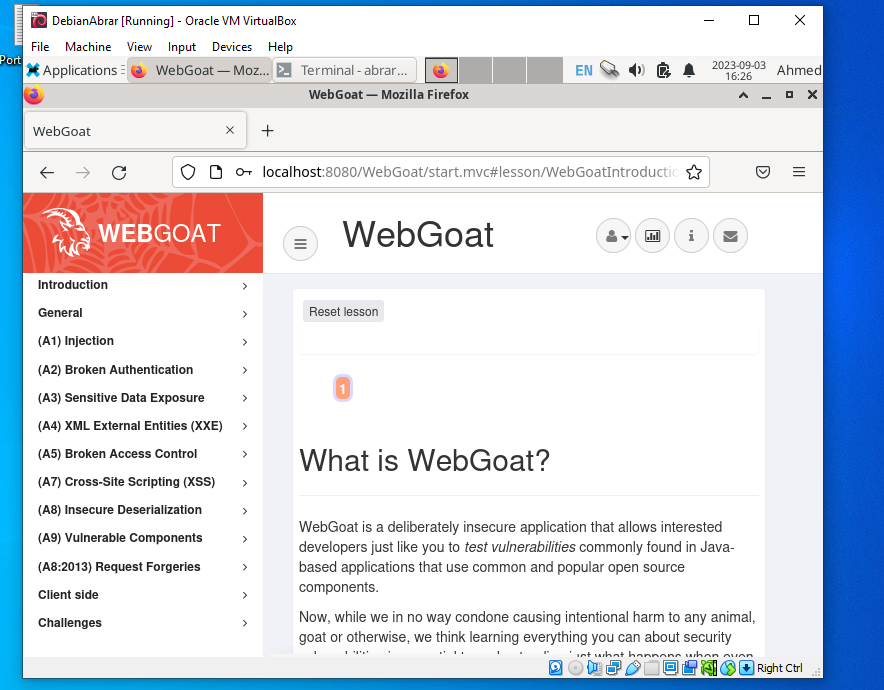
1. Filter database inputs.

2. Restrict database code.

3. Restrict database access.

4. Maintain applications and databases.

5. Monitor application and database inputs and communications.

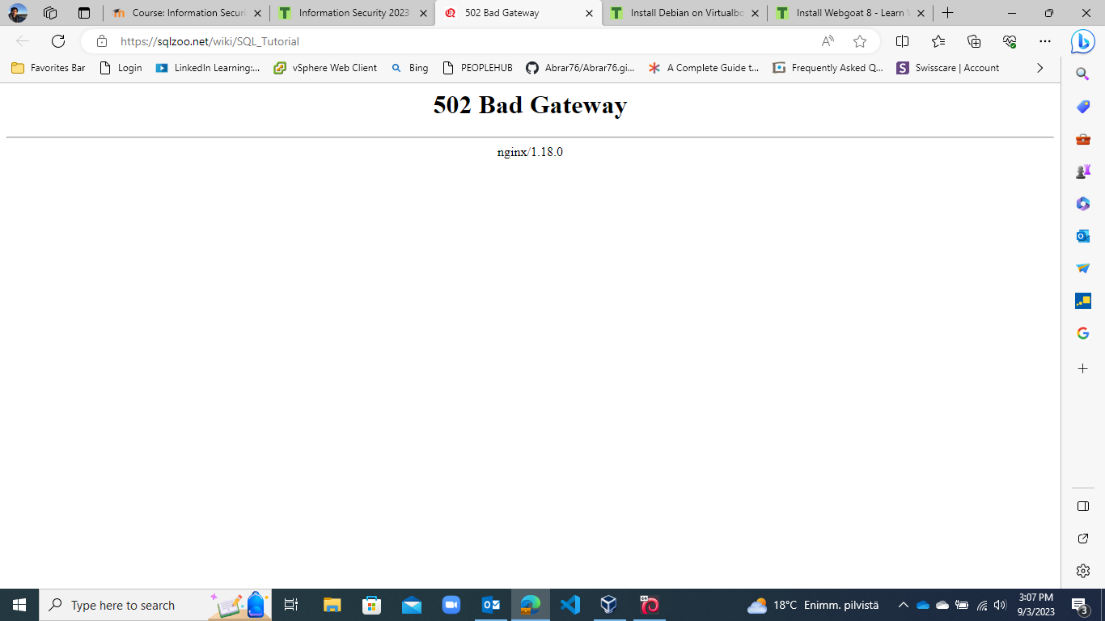
**a) Goat. Install webgoat**

**b) F12. Solve Webgoat 8: General: Developer tools**

**c) Not outdated. Update all operating system and all applications in your Linux.**

**d) Sequel. Solve SQLZoo:**

0 SELECT basics 2 SELECT from World, from first two subtasks



**e) Johnny tables. Solve Portswigger Labs: Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data.**